Taxxo v1.0: an R package for integrating and automatising tools for prokaryotes taxonogenomics

Gregorio IRAOLA

January 6, 2017

FULL USER MANUAL

Contents

1 Overview

Taxonogenomics has emerged as a incipient branch within polyphasic taxonomy of prokaryotes. This considers the whole genome sequences (or a subset of their components) as the ultimate source of information for species delimitation. Under this premise many analytical approaches have been proposed, but most of them are not integrated and automatised in software tools. Here we adapt the most widely used tools for bacterial taxonogenomics into an R package, providing a framework that can be easily integrated into computational biology pipelines.

Taxxo provides several different analytical tools, including 16S and 23S genes similarity and phylogenetics, bacterial and archaeal universal proteins, Average Nucleotide Identity (ANI), Average Amino Acid Identity (AAI) and Average Genome Identity of Orthologous Sequences (AGIOS). These tools can be ran independently or in parallel for any desired set of assembled genomes.

Taxxo is not algorithmically novel, nor provides any novel way of measuring the relatedness of prokaryotes genomes. By the contrary, what makes Taxxo useful is the integration and automatisation of state-of-the-art tools used in taxogenomic analyses, which allow to easily integrate them in computational biology pipelines and to manage and visualise the results in standard formats.

Developer:

Gregorio IRAOLA[1] - giraola@pasteur.edu.uy

[1]Unidad de Bioinformática, Institut Pasteur Montevideo, Uruguay.

Citation:

`https://github.com/giraola/taxxo`

2 Installation

The first stable version of Taxxo (v1.0) is currently available at **GitHub** repository via `https://github.com/giraola/taxxo`.

```
# INSTALL TAXXO

library(devtools)
install_github('/giraola/taxxo')
```

2.1 Platforms

Currently Taxxo works on **Mac** (tested on El Capitan and Sierra) and **Linux** (tested on Fedora and Ubuntu). A Windows version is not in the agenda.

2.2 External dependencies

Taxxo depends on a set of other software packages, the binary files are included within Taxxo and there is no need of installing any of them separately:

- **BLAST+ -** Basic Local Alignment Search Tool [1]. Taxxo uses blastn, blastp, tblastx, makeblastdb and blastdbcmd.
- **Prodigal -** Prokaryotic dynamic programming gene-finding algorithm [2].
- **HMMER3.1 -** Searching homologous sequences using hidden Markov models (HMMs) [3]. Taxxo uses hmmsearch and hmmbuild.
- **barrnap -** Basic rRNA prediction [4].

2.3 Internal dependencies

Taxxo also uses a handful of tools for particular tasks within R which depend on the following packages:

- **msa -** Multiple Sequence Alignment [5]. Taxxo runs Clustal Omega multiple sequence aligner from inside R using the 'msa' package [5].
- **phangorn -** Phylogenetics [6]. NJ and ML tree reconstruction.
- **seqinr -** DNA and protein sequences in/out and formatting [7].
- **doMC -** Parallel computing.

3 Analysis of ribosomal RNA genes (rrna)

3.1 Description

The sequence and phylogenetic analysis of ribosomal RNA genes, in particular the small subunit 16S, has been the paradigm for genotypic identification of prokaryotic species. Taxxo provides a specific function called **rrna()** that automatically predicts, extracts and aligns 16S or 23S sequences from archaeal or bacterial assembiles.

3.2 Usage

```
# rrna()

rrna(path = 'path/to/test_genomes',
      pattern = '.fna',
      outdir = "./rrna_result",
      subunit = '16S',
      kingdom = 'bacteria',
      align = TRUE,
      distance = TRUE,
      phylogeny = TRUE)

# Arguments

# path - the path where input assemblies in FASTA format are stored.
# pattern - is a pattern to recognise the assembly files.
# outdir - the output directory that will be created with the results.
# subunit - takes '16S', '23S' or 'both' setting the desired analysis.
# kingdom - takes 'bacteria' or 'archaea'.
# align - a logical indicating if alignment should be performed or not.
# distance - a logical indicating if a distance matrix is calculated.
# phylogeny - a logical indicating if a NJ phylogeny is constructed.
```

3.3 Output

The **rrna()** function produces a maximum of 5 types of output files. The suffix 16S is taken as example here. All output files are automatically written inside the specified output directory:

- ***.16S.fasta**: Are single FASTA file containing the extracted sequences for each analysed genome.
- **all.16S.fasta**: A multi FASTA file containing all the sequences for all analysed genomes.
- **alignment.16S.fasta**: A multiple sequence alignment including all recovered sequences (if **align=TRUE**).

- **distance_matrix_16S.Rdata**: An object loadable with **load()** function displaying a p-distance matrix (if **distance=TRUE**).
- **NJ.16S.tree.nwk**: A Newick format phylogenetic tree produced with the NJ algorithm over the alignment of all sequences (if **phylogeny=TRUE**).

4 Protein coding genes detection (prodigal)

4.1 Description

Some tools implemented in Taxxo require to analyse protein coding genes. For providing this input to these tools, the function **prodigal()** implements the Prodigal software for gene prediction [2].

4.2 Usage

```
# prodigal()

prodigal(path = '/path/to/genomes',
   pattern = '.fna',
   outdir = 'prodigal_result',
   proc = 2)

# Arguments

# path - the path where input assemblies in FASTA format are stored.
# pattern - is a pattern to recognise the assembly files.
# outdir - the output directory that will be created with the results.
# proc - the number of CPUs for parallel computing.
```

4.3 Output

This function outputs 4 different types of files per analysed genome:

- ***.faa**: FASTA files in amino acids containing the protein coding genes.
- ***.ffn**: FASTA files in nucleotides containing the protein coding genes.
- ***.out**: Protein coding genes in tabular format.
- ***.log**: A log file summarizing the Prodigal run.

5 Analysis of 40 universal proteins (uprot)

5.1 Description

Considering that the ribosomal RNA genes can show sub-optimal phylogenetic signal for some organisms, various efforts to find a good set of protein coding genes useful as universal phylogenetic markers have been developed [8, 9]. This lead to the identification of a 40 universal phylogenetic markers which are single copy genes and occur in the vast majority of known organisms, and have been successfully used to reconstruct high resolution phylogenies [8]. Within Taxxo, the function **uprot()** automatically identifies, extracts, concatenates and aligns this set of 40 universal protein markers from a set of annotated genomes.

5.2 Usage

```
# uprot()

uprot(path = '/path/to/prodigal_result',
      pattern = '.faa',
      outdir = './uprot_result',
      align = TRUE,
      phylogeny = TRUE,
      proc=2)

# Arguments

# path - the path to the input AAs annotations in FASTA format.
# pattern - is a pattern to recognise the annotation files.
# outdir - the output directory that will be created with the results.
# align - a logical indicating if alignment should be performed or not.
# phylogeny - a logical indicating if a NJ phylogeny is constructed.
# proc - is the number of CPUs for parallel computing.
```

5.3 Output

The **uprot()** function produces a maximum of 5 types of output files. All output files are automatically written inside the specified output directory:

- **COG*.uprot.faa**: Are multi FASTA files containing the extracted sequences for each analysed genome, for each of the 40 universal proteins designated with their corresponding COG identifiers.
- **COG*.uprot.ali**: Are multi FASTA files containing the aligned sequences for each of the 40 universal proteins designated with their corresponding COG identifiers (if **align=TRUE**).

- **universal_proteins.ali**: A multiple sequence alignment including all concatenated sequences (if **align=TRUE**).
- **presence_absence.Rdata**: An object loadable with **load()** function displaying a presence/absence matrix with genomes in rows and proteins in columns.
- **NJ.uprot.tree.nwk**: A Newick format phylogenetic tree produced with the NJ algorithm over the alignment of concatenated sequences (if **phylogeny=TRUE**).

6 Analysis of 37 archaeal ribosomal proteins (aprot)

6.1 Description

The function **aprot()** follows exactly the same idea than **uprot()**, but the difference is that the set of protein coding genes are optimised for **Archaea**, allowing to specifically work with this domain of prokaryotes. The archaeal protein set was obtained from TIGRFAMs [10] GenProp0831, which describes an archaeal core gene set occurring one per genome.

6.2 Usage

```
# aprot()

aprot(path = '/path/to/prodigal_result',
      pattern = '.faa',
      outdir = './aprot_result',
      align = TRUE,
      phylogeny = TRUE,
      proc=2)

# Arguments

# path - the path to the input AAs annotations in FASTA format.
# pattern - is a pattern to recognise the annotation files.
# outdir - the output directory that will be created with the results.
# align - a logical indicating if alignment should be performed or not.
# phylogeny - a logical indicating if a NJ phylogeny is constructed.
# proc - is the number of CPUs for parallel computing.
```

6.3 Output

The **aprot()** function produces a maximum of 5 types of output files. All output files are automatically written inside the specified output directory:

- **TIGR*.aprot.faa**: Are multi FASTA files containing the extracted sequences for each analysed genome, for each of the archaeal proteins designated with their corresponding TIGRFAMs identifiers.
- **TIGR*.aprot.ali**: Are multi FASTA files containing the aligned sequences for each of the archeal proteins designated with their corresponding TIGRFAMs identifiers (if **align=TRUE**).
- **archaeal_proteins.ali**: A multiple sequence alignment including all concatenated sequences (if **align=TRUE**).

- **presence_absence.Rdata**: An object loadable with **load()** function displaying a presence/absence matrix with genomes in rows and proteins in columns.
- **NJ.aprot.tree.nwk**: A Newick format phylogenetic tree produced with the NJ algorithm over the alignment of concatenated sequences (if **phylogeny=TRUE**).

7 Single phylogenetic marker gene (mgene)

7.1 Description

For many taxa, the single marker genes has been investigated and provide improved phylogenetic and taxonomic resolution. A good example is the *gyrA* gene in the genus *Helicobacter*, whose phylogenetic resolution is better than 16S [11]. Given a multi FASTA file containing the marker gene sequences, the function **mgene()** automatically looks for this gene in a set of genomes and retrieves them in FASTA format.

7.2 Usage

```
# mgene()

mgene(path = '/path/to/prodigal_result',
      pattern = '.faa',
      outdir = './mgene_result',
      marker='/path/to/marker.faa',
      align = TRUE,
      phylogeny = TRUE,
      proc=2)

# Arguments

# path - the path to the input AAs annotations in FASTA format.
# pattern - is a pattern to recognise the annotation files.
# outdir - the output directory that will be created with the results.
# marker - the multi FASTA file containing the marker gene sequences.
# align - a logical indicating if alignment should be performed or not.
# phylogeny - a logical indicating if a NJ phylogeny is constructed.
# proc - is the number of CPUs for parallel computing.
```

7.3 Output

The **mgene()** function produces a maximum of 6 types of output files:

- ***mgene.ali**: The multi FASTA file of the aligned marker gene sequences.
- ***mgene.hmm**: The HMM built from the aligned marker gene sequences.
- **presence_absence_*.Rdata**: An object to **load()** displaying a presence/absence of the marker gene in each genome.
- ***_all.faa**: A multi FASTA file containing the marker gene sequences recovered from the genomes.

- ***_all.ali**: A multi FASTA file containing the aligned marker gene sequences recovered from the genomes (if **align=TRUE**).
- **NJ.mgene.tree.nwk**: A Newick format phylogenetic tree produced with the NJ algorithm over the alignment of concatenated sequences (if **phylogeny=TRUE**).

8 Average Nucleotide Identity (anib)

8.1 Description

The concept of Average Nucleotide Identity (ANI) was first described by Goris *et al.* (2007) [12], as a genomic measure to substitute the DNA-DNA hibridization (DDH) techniques previously used as the gold standard for prokaryotes species delimitation. With the increasing availability of genomic sequences, the ANI calculation is currently used as a routine tool for species delimitation. Here the function **anib()** implements the original algorithm based on BLAST searches.

8.2 Usage

```
# anib()

anib(path = '/path/to/genomes/',
      pattern = '.fna',
      outdir = './anib_result',
      reference='all',
      proc=2)

# Arguments

# path - the path to the input genome assemblies in FASTA format.
# pattern - is a pattern to recognise the genome files.
# outdir - the output directory that will be created with the results.
# proc - is the number of CPUs for parallel computing.
# reference - if set to "all" (default), all vs. all genome comparisons
    are performed. If set to the name of any genome file, all vs. this
    reference are performed.
```

8.3 Output

The **anib()** function produces a strictly one output file: **anib_result.Rdata**. It can be accessed using **load()** and consists in a 3-column data frame where columns 1 and 2 correspond to each possible pairs of genomes and the column 3 stores the ANIb value.

9 Average Amino acid Identity (aai)

9.1 Description

The genetic relatedness between a pair of genomes can be also measured by the average amino acid identity (AAI) of all conserved genes between the two genomes as computed by the BLASTp algorithm [13]. Here the function **aai()** implements this calculation.

9.2 Usage

```
# aai()

aai(path = '/path/to/prodigal_result/',
      pattern = '.faa',
      outdir = './aai_result',
      reference='all',
      proc=2)

# Arguments

# path - the path to the input annotations files in FASTA format.
# pattern - is a pattern to recognise the annotation files.
# outdir - the output directory that will be created with the results.
# proc - is the number of CPUs for parallel computing.
# reference - if set to "all" (default), all vs. all genome comparisons
    are performed. If set to the name of any genome file, all vs. this
    reference are performed.
```

9.3 Output

The **aai()** function produces strictly one output file: **aai_result.Rdata**. It can be accessed using **load()** and consists in a 3-column data frame where columns 1 and 2 correspond to each possible pairs of genomes and the column 3 stores the AAI value.

10 Average Genetic Identity of Orthologous Sequences (agios)

10.1 Description

The AGIOS calculation is performed by first identifying with BLASTp the set of orthologous proteins between pairs of genomes using a coverage threshold of 50% and an amino acid identity threshold of 30%. Then, the average percentage of nucleotide sequence identity is determined between these orthologous genes [14].

10.2 Usage

```
# agios()

aai(path = '/path/to/prodigal_result/',
      pattern = '.faa',
      outdir = './agios_result',
      reference='all',
      proc=2)

# Arguments

# path - the path to the input annotations files in FASTA format.
# pattern - is a pattern to recognise the annotation files.
# outdir - the output directory that will be created with the results.
# proc - is the number of CPUs for parallel computing.
# reference - if set to "all" (default), all vs. all genome comparisons
    are performed. If set to the name of any genome file, all vs. this
    reference are performed.
```

10.3 Output

The **agios()** function produces strictly one output file: **agios_result.Rdata**. It can be accessed using **load()** and consists in a 3-column data frame where columns 1 and 2 correspond to each possible pairs of genomes and the column 3 stores the AGIOS value.

11 References

[1] Christiam Camacho, George Coulouris, Vahram Avagyan, Ning Ma, Jason Papadopoulos, Kevin Bealer, and Thomas L Madden. Blast+: architecture and applications. *BMC bioinformatics*, 10(1):1, 2009.

[2] Doug Hyatt, Gwo-Liang Chen, Philip F LoCascio, Miriam L Land, Frank W Larimer, and Loren J Hauser. Prodigal: prokaryotic gene recognition and translation initiation site identification. *BMC bioinformatics*, 11(1):1, 2010.

[3] L Steven Johnson, Sean R Eddy, and Elon Portugaly. Hidden markov model speed heuristic and iterative hmm search procedure. *BMC bioinformatics*, 11(1):1, 2010.

[4] Torsten Seemann. barrnap 0.7: rapid ribosomal rna prediction. *https://github.com/Victorian-Bioinformatics-Consortium/barrnap*, 2013.

[5] Ulrich Bodenhofer, Enrico Bonatesta, Christoph Horejs-Kainrath, and Sepp Hochreiter. msa: an r package for multiple sequence alignment. *Bioinformatics*, 31(24):3997–3999, 2015.

[6] Klaus Peter Schliep. phangorn: phylogenetic analysis in r. *Bioinformatics*, 27(4):592–593, 2011.

[7] D. Charif, J. Thioulouse, J. R. Lobry, and G. Perriere. Online synonymous codon usage analyses with the ade4 and seqinR packages. *Bioinformatics*, 21(4):545–547, Feb 2005.

[8] Francesca D Ciccarelli, Tobias Doerks, Christian Von Mering, Christopher J Creevey, Berend Snel, and Peer Bork. Toward automatic reconstruction of a highly resolved tree of life. *science*, 311(5765):1283–1287, 2006.

[9] Rotem Sorek, Yiwen Zhu, Christopher J Creevey, M Pilar Francino, Peer Bork, and Edward M Rubin. Genome-wide experimental determination of barriers to horizontal gene transfer. *Science*, 318(5855):1449–1452, 2007.

[10] Daniel H Haft, Jeremy D Selengut, Roland A Richter, Derek Harkins, Malay K Basu, and Erin Beck. Tigrfams and genome properties in 2013. *Nucleic acids research*, 41(D1):D387–D395, 2013.

[11] Armelle Ménard, Alice Buissonnière, Valérie Prouzet-Mauléon, Elodie Sifré, and Francis Mégraud. The gyra encoded gene: A pertinent marker for the phylogenetic revision of helicobacter genus. *Systematic and applied microbiology*, 39(2):77–87, 2016.

[12] Johan Goris, Konstantinos T Konstantinidis, Joel A Klappenbach, Tom Coenye, Peter Vandamme, and James M Tiedje. Dna–dna hybridization values and their relationship to whole-genome sequence similarities. *International journal of systematic and evolutionary microbiology*, 57(1):81–91, 2007.

[13] Konstantinos T Konstantinidis and James M Tiedje. Towards a genome-based taxonomy for prokaryotes. *Journal of bacteriology*, 187(18):6258–6264, 2005.

[14] Dhamodharan Ramasamy, Ajay Kumar Mishra, Jean-Christophe Lagier, Roshan Padhmanabhan, Morgane Rossi, Erwin Sentausa, Didier Raoult, and Pierre-Edouard Fournier. A polyphasic strategy incorporating genomic data for the taxonomic description of novel bacterial species. *International journal of systematic and evolutionary microbiology*, 64(2):384–391, 2014.